Q1:

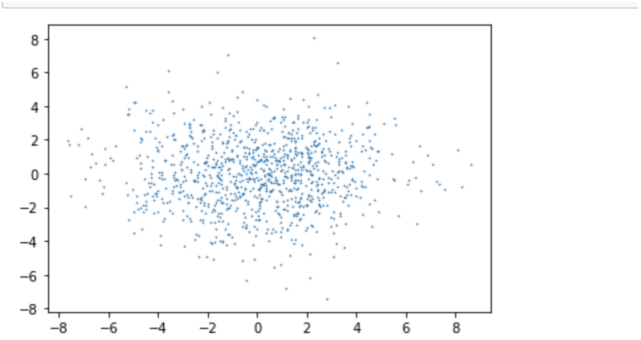
Correlation between sensation seeking behavior and movie experience is below : (heatmap)

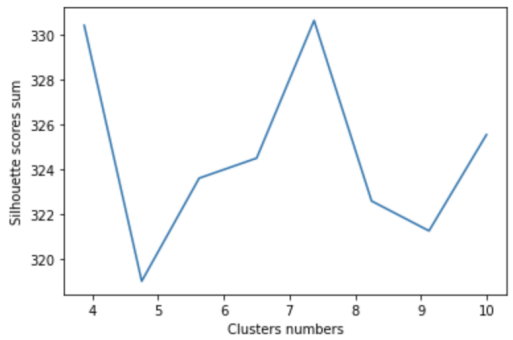
[93]:

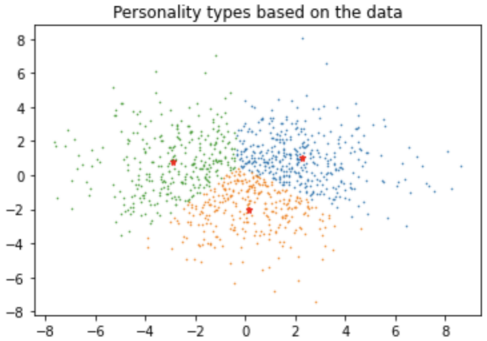
```
db = pd.concat([df.iloc[:, 400:420], df.iloc[:, 464:474]], axis=1)

db = db.dropna()
corr = db.corr()
plt.plot(figsize = (15, 10))
sns.heatmap(corr, annot = True)
plt.show()
```

Q2:

Applying a PCA dimension reduction method on the personality data to reduce it into dimentions. Then I visualize the personality. I chose the optimal number of the clusters using the silhouette analysis method and apply a K-means cluster method to get the personality type. Then I compute the means using 3 clusters and plot the data.







Q3: Yes.

First, I count the ratings recived from each movies into two categories. Then I set the null hypo as movies that are more popular rated are NOT higher than movies that are less popular. Since from the p value I got, $p<0.05$. Then, we reject the H0, movies that are more popular rated are higher than movies that are less popular!

Q4: NO!

First, I set the null hypo as there is no differences between the ratings of "Shrek 2002" from famles and males. Also, I categorized the number of femals and males' ratings. From the mann whitney u test(two sided), we got the p value > 0.05, then I failed to reject the H0. Which means, there is not a difference in the ratings of "Shrek 2001" between males and female.

791 females rated it, the average is 3.16.

254 males rated it, the average is 3.08.

Q5: NO!

I first set the H0 as people who are only children do NOT enjoy 'The Lion King (1994)' more than people with siblings. Then, from the mann whitney u test(one-tailed), the p value is 0.98 > 0.05. Then, I falied to reject the H0, which means people who are only children do NOT enjoy 'The Lion King (1994)' more than people with siblings.

Q6: NO!

I first set the H0 as people who like to watch movies socially do NOT enjoy 'The Wolf of Wall Street (2013)' more than those who prefer to watch them alone. Then, from the mann whitney u test(one-tailed), the p value is 0.94 > 0.05. Then, I falied to reject the H0, which means people who like to watch movies socially do NOT enjoy 'The Wolf of Wall Street (2013)' more than those who prefer to watch them alone.

Q7:

We define the H0 as the movies in a particular group that are not of inconsistent quality. And I use the Kruskal Wallis test on all the movies from the group. The results showed that the "Harry Potter" and “Pirates of the Caribbean” have a p-value>0.05, for all others less than0.05. Hence I reject the H0 for all the other movies. However, I fail to reject the H0 for "Harry Potter" and"Pirates of the Caribbean”. Therefore, “Harry Potter” and “Pirates of the Caribbean” are of consistent quality while all others movies are of inconsistent quality, as experienced by viewers.

```
Star Wars 6.940162236984522e-40
Harry Potter 0.11790622831256074
The Matrix 1.7537323830838066e-09
Indiana Jones 1.020118354785894e-11
Jurassic Park 1.8492328391686058e-11
Pirates of the Caribbean 0.035792727694248905
Toy Story 7.902234665149812e-06
Batman 4.1380499020034183e-19
```

Q8:

I deleted the missing values and split the data as train and test to avoid overfitting. I trained the multicple regression model and test training and test dataset. I got the max RMSE for movie “The Doom Generation(1995)” and Min RMSE for movie”X-men2(2003)”.

```
ssb_zscore = scp.zscore(np.delete(ssb, np.where(np.isnan(ssb))[0], 0))
mer_zscore = scp.zscore(np.delete(mer, np.where(np.isnan(mer))[0], 0))
pqu_zscore = scp.zscore(np.delete(pqu, np.where(np.isnan(pqu))[0], 0))
pca = PCA()
ssb_PCA = pca.fit(ssb_zscore)
mer_PCA = pca.fit(mer_zscore)
pqu_PCA = pca.fit(pqu_zscore)
def multiple_regression(x, y):
    print(len(x[0]))
    cols = np.arange(0,len(x[0]))
    rmse = np.array([])
    for i in range(len(y[0])):
        nx = x
        ny = y[:,i]
        isnan_data = np.unique(np.concatenate((np.where(np.isnan(x))[0],np.where(np.isnan(ny))[0]),0))
        predictor = np.delete(x, isnan_data, 0)
        prediction = np.delete(ny, isnan_data, 0)
        predictor[:,cols] = pqu_PCA.fit_transform(predictor[:,cols])

        x_train, x_test, y_train, y_test  = train_test_split(predictor, prediction, test_size = 0.2, random_state = 42)
        regr = LinearRegression()
        regr.fit(x_train, y_train)

        rmse = np.append(rmse, np.sqrt(mean_squared_error(y_test, regr.predict(x_test))))
    return rmse
x=data[:,420:464]
y=data[:, :400]
rmse = multiple_regression(x,y)
max_rmse = max(rmse)
min_rmse = min(rmse)
for i in range(len(rmse)):
    if rmse[i] == min_rmse:
        print("min RMSE:%f\t movie:%s"%( rmse[i], header[i]))
    if rmse[i] == max_rmse:
        print("max RMSE:%f\t movie:%s"%( rmse[i], header[i]))
```

```
44
max RMSE:31552306320990.710938   movie:The Doom Generation (1995)
min RMSE:0.718424        movie:X-Men 2 (2003)
```

Q9:

I did the same thing for Q9. I deleted the missing values and split the data as train and test to avoid overfitting. I trained the multicple regression model and test training and test dataset. I got the max RMSE for movie “Shawshank Redemption” and Min RMSE for movie”Ran(1985)”.

```
#Q9
def multiple_regression(x, y):

    rmse = np.array([])
    for i in range(len(y[0])):
        nx = x
        ny = y[:,i]
        isnan_data = np.unique(np.concatenate((np.where(np.isnan(x))[0],np.where(np.isnan(ny))[0]),0))
        predictor = np.delete(x, isnan_data, 0)
        prediction = np.delete(ny, isnan_data, 0)

        x_train, x_test, y_train, y_test  = train_test_split(predictor, prediction, test_size = 0.33, random_state = 42
        regr = LinearRegression()
        regr.fit(x_train, y_train)

        rmse = np.append(rmse, np.sqrt(mean_squared_error(y_test,regr.predict(x_test))))
    return rmse


x=data[:, 475:477]
y=data[:, :400]
rmse = multiple_regression(x,y)


max_rmse = max(rmse)
min_rmse = min(rmse)
for i in range(len(rmse)):

    if rmse[i] == min_rmse:
        print("min RMSE:%f\t movie:%s"%( rmse[i], header[i]))

    if rmse[i] == max_rmse:
        print("max RMSE:%f\t movie:%s"%( rmse[i], header[i]))
```

```
min RMSE:0.731987        movie:The Shawshank Redemption (1994)
max RMSE:1.437749        movie:Ran (1985)
```

Q10:

I did the same thing for Q10. I deleted the missing values and split the data as train and test to avoid overfitting. I trained the multicple regression model and test training and test dataset. I got the max RMSE for movie “Equilibrium (2002)” and Min RMSE for movie “The Lion King (1994)”.

```
# Q10
def multiple_regression(x, y):
    print(len(x[0]))
    cols = np.arange(0,20)
    ssb_cols = np.arange(20,64)
    mer_cols = np.arange(64,74)
    rmse = np.array([])
    for i in range(len(y[0])):
        nx = x
        ny = y[:,i]
        isnan_data = np.unique(np.concatenate((np.where(np.isnan(x))[0],np.where(np.isnan(ny))[0]),0))
        predictor = np.delete(x, isnan_data, 0)
        prediction = np.delete(ny, isnan_data, 0)
        predictor[:,cols] = pqu_PCA.fit_transform(predictor[:,cols])
        predictor[:,ssb_cols] = ssb_PCA.fit_transform(predictor[:,ssb_cols])
        predictor[:,mer_cols] = mer_PCA.fit_transform(predictor[:,mer_cols])
        x_train, x_test, y_train, y_test  = train_test_split(predictor, prediction, test_size = 0.2, random_state = 42)
        regr = LinearRegression()
        regr.fit(x_train, y_train)
        rmse = np.append(rmse, np.sqrt(mean_squared_error(y_test, regr.predict(x_test))))
    return rmse
x=data[:, 400:477]
y=data[:, :400]
rmse = multiple_regression(x,y)
max_rmse = max(rmse)
min_rmse = min(rmse)
for i in range(len(rmse)):
    if rmse[i] == min_rmse:
        print("min RMSE:%f\t movie:%s"%( rmse[i], header[i]))
    if rmse[i] == max_rmse:
        print("max RMSE:%f\t movie:%s"%( rmse[i], header[i]))
```

```
77
max RMSE:101.832793      movie:Equilibrium (2002)
min RMSE:0.676393        movie:The Lion King (1994)
```